Attorney's Do[redacted] No.: 10454-022001 / P-4190-4

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: PROBABILISTIC ALERT CORRELATION

APPLICANT: ALFONSO DE JESUS VALDES AND KEITH SKINNER

David L. Feigenbaum
Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Tel.: (617) 542-5070
Fax: (617) 542-8906

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL 485 518 474 US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

August 31, 2001
Date of Deposit

Samantha Bell
Signature

Samantha Bell
Typed or Printed Name of Person Signing Certificate

Probabilistic Alert Correlation

REFERENCE TO GOVERNMENT FUNDING

This invention was made with Government support under contract numbers F30602-99-C-0149 and N66001-00-C-8058 awarded by DARPA. The Government has certain rights in this invention.

REFERENCE TO RELATED APPLICATION

This is a continuation-in-part of U.S. patent application serial number 09/711,323, filed November 9, 2000 and entitled "Sensor and Alert Correlation in Intrusion Detection Systems," which is a continuation-in-part of U.S. patent application serial number 09/653,066, filed September 1, 2000 and entitled "Methods for Detecting and Diagnosing Abnormalities Using Real-Time Bayes Networks." Both of these patent applications are incorporated herein by reference. This application also claims priority under 35 USC §119(e) from co-pending Provisional patent application number 60/287,514, filed March 23, 2001, naming Alfonso de Jesus Valdes and Keith M. Skinner as inventors and entitled "Probabilistic Alert Correlation," which is incorporated herein by reference.

TECHNICAL FIELD

The invention relates generally to intrusion detection, and more specifically to sensor and alert correlation in intrusion detection systems.

BACKGROUND

There are three main types of intrusion detection systems currently used to detect hacker attacks on computer networks: signature analysis systems, statistical analysis systems, and systems based on probabilistic reasoning. Signature analysis systems compare current data traffic patterns with stored traffic patterns representing the signature or profile of various types of hacker attacks. These systems generate an alert if the pattern of traffic received by the network matches one of the stored attack patterns.

Statistical analysis systems compare current data traffic patterns with statistical profiles of previous traffic patterns. These systems generate an alert if a current traffic

pattern is significantly different from a stored profile of "normal" traffic. Examples of both statistical- and signature-based intrusion detection systems are described in Porras, et al., "*Live Traffic Analysis of TCP/IP Gateways*," Internet Society's Networks and Distributed Systems Society Symposium, March 1998.

Examples of intrusion detection systems based on probabilistic reasoning are described in U.S. patent application serial number 09/653,066 entitled "Methods for Detecting and Diagnosing Abnormalities Using Real-Time Bayes Networks." In one such system, a Bayes network is established that includes models (called "hypotheses") that represent both normal traffic and attack traffic received by a computer network. Traffic actually received by the network is examined in real-time to identify its relevant characteristics or features, such as volume of data transfer, number of erroneous connection requests, nature of erroneous connection requests, ports to which connections are attempted, etc. Information about these relevant features is then provided to the Bayes network, which calculates a system belief (a probability) that the current network traffic is either normal traffic or attack traffic.

A typical intrusion detection system may include one or more sensors that monitor network traffic in the manner discussed above, and one or more other sensors that monitor network resources. A system operator or network administrator (usually a person) reviews all of the alerts generated by the system.

A major problem with existing intrusion detection systems is that they often provide misleading, incomplete, or low-quality information to the system operator; this may make it impossible to take proper steps to protect the network. For example, during a large-scale hacker attack each of the system's sensors may generate hundreds of alerts. Although each alert may be accurate, the sheer number of alerts could easily overwhelm the system operator. Moreover, false alarms can be triggered by normal traffic directed towards a malfunctioning network resource, such as a server. These false alarms could distract the system operator from alerts triggered by actual hacker attacks. Finally, most intrusion detection systems are unable to detect low-level attacks such as port sweeps, in which hackers slowly "probe" a network to discover its structure and weaknesses.

SUMMARY

This invention uses probabilistic correlation techniques to increase sensitivity, reduce false alarms, and improve alert report quality in intrusion detection systems. In one preferred embodiment, an intrusion detection system includes at least two sensors to monitor different aspects of a computer network, such as a sensor that monitors network traffic and a sensor that discovers and monitors available network resources. The sensors are correlated in that the belief state of one sensor is used to update or modify the belief state of another sensor. For example, a network resource sensor may detect that a server is malfunctioning. This information is used to modify the belief state of one or more network traffic sensors so that normal network traffic directed towards the malfunctioning server does not trigger a false alarm.

In another example, a network resource sensor transmits its knowledge of network structure to a network traffic sensor. This information is used to modify the belief state of the network traffic sensor so that an attempt to communicate with a non-existent resource appears to be suspicious. By allowing different sensors to share information, the system's sensitivity to low-level attacks can be greatly increased, while at the same time greatly reducing the number of false alarms.

In another embodiment, probabilistic correlation techniques are used to organize alerts generated by different types of sensors. By comparing features of each new alert with features of previous alerts, rejecting a match if a feature fails to meet a minimum similarity value, and adjusting the comparison by an expectation that certain feature values will or will not match, the alerts can be grouped in an intelligent manner. The system operator may then be presented with a few groups of related alerts, rather than lots of individual alerts from multiple sensors.

Other features, objects, and advantages of the invention will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

Figure 1 shows a preferred intrusion detection system with coupled sensors.

Figure 2 is a flowchart showing a preferred method for coupling sensors.

Figure 3 shows a preferred intrusion detection system with coupled sensors and an alert correlation device.

Figure 4 is a flowchart showing a preferred method for grouping alerts into classes of related alerts.

Figure 5 is an example of an incident class similarity matrix as used in one embodiment.

Figure 6 is an example of a display showing a correlated attack report.

Figure 7 is an example of a display showing raw sensor alerts.

Figure 8 is an example of a display showing the results of a preferred alert fusion process.

Figure 9 is an example of a display showing the results of a preferred incident matching process.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

I. Introduction

In preferred embodiments, intrusion detection systems for computer networks include sensors that monitor both network traffic and network resources. Correlation techniques are used to increase system sensitivity, reduce false alarms, and organize alerts into related groups. The sensor and alert correlation techniques described below may be used in intrusion detection systems that include any type or mix of sensors. However, probabilistic sensors similar to those described in U.S. Patent Application Serial Number 09/653,066 may provide advantages over other types of sensors.

Bayes networks are preferably used to perform the probabilistic reasoning functions described below. As is more fully described in U.S. Patent Application Serial Number 09/653,066, Bayes networks include models (called "hypotheses") that represent some condition or state; probabilistic reasoning methods are used to determine a belief that a current observation corresponds to one of the stored hypotheses. Before a Bayes network performs an analysis, an initial belief (called a "prior probability") regarding the various hypotheses is established. During the analysis, a new belief based on actual observations is established, and the prior probability is adjusted accordingly. In the context of the embodiments described here, a relevant Bayes hypothesis might be "there is a stealthy portsweep attack against the computer network." Belief in the portsweep attack hypothesis

would be strengthened if attempts to connect to several invalid or nonexistent ports are observed.

II. Intrusion Detection with Correlated Sensors

As was discussed above, intrusion detection systems typically have several independent sensors that gather different types of information. In preferred embodiments of this invention, the information gathered by the various sensors may be shared to improve overall performance of the intrusion detection system.

In a preferred intrusion detection system 100 shown in Figure 1, sensors are coupled so that the belief state of one sensor (such as network resource sensor 103) affects the belief state of another sensor (such as network traffic sensor 101). Each sensor may then transmit alerts to a system operator or network administrator 105. These sensors need not be probabilistic sensors; for example, a sensor that keeps track of the number of servers in a network wouldn't need to generate a probabilistic output.

Figure 2 illustrates a preferred method by which sensors are coupled or correlated. At step 201, a first (preferably probabilistic) sensor receives all or part of the belief state of a second (not necessarily probabilistic) sensor. The belief state of the second sensor may indicate an apparent normal, degraded, or compromised state of a monitored system resource, the existence or validity of supported services, or any other relevant belief state held by a sensor in an intrusion detection system. At step 203, a prior belief state of the first sensor is adjusted, the adjustment based at least in part on the second sensor's belief state. For example, the prior belief state of the first sensor or may be adjusted so that an erroneous transaction with a damaged or compromised network resource does not generate an alert. In another example, the prior belief state of the first sensor may be adjusted so that an attempted communication with a system server or resource that does not exist appears to be suspicious. By allowing different sensors to share information, the system's sensitivity to low-level attacks can be greatly increased, while at the same time greatly reducing the number of false alarms.

III. Alert Correlation

In another embodiment, probabilistic correlation techniques are used to organize alerts into classes or groups of related alerts. By comparing features of each new alert with features of previous alerts, and adjusting the comparison by an expectation that certain feature values will or will not match, the alerts can be grouped in an intelligent manner. Alerts may be grouped or organized in several different ways, depending on both the type of an attack and the structures of the intrusion detection system and the monitored network. For example, all alerts related to a specific attack (such as a denial of service attack) may be grouped together. Or, all alerts related to the various stages of a staged attack may be grouped together to allow the system operator to view the progression of a staged attack.

A. Alert Correlation Devices

Figure 3 shows a preferred intrusion detection system 300 that includes a network traffic sensor 301 and network resource sensor 303. Intrusion detection system 300 may include any number of sensors, and the belief states of the sensors may be correlated in the manner discussed above. Alerts generated by sensors 301 and 303 are provided to an alert correlation device 305, which uses probabilistic reasoning (preferably Baysian) techniques to organize alerts into classes of related alerts. These alert classes are then provided to a system operator or network administrator 307.

B. Measurement of Similarity

To group alerts in an intelligent manner, it is necessary to define and determine the degree of similarity between a new alert and an existing alert or alert class. This measurement of similarity is preferably calculated by comparing similarity among shared features (also called "measures") of the alert and alert class. Examples of features to consider when assessing similarity between a new alert and an existing alert class include source IP address, destination IP address and port, type of alert, type of attack, etc. The nature of an alert may change the expectation of which features should be similar. It is therefore important to define a measurement of similarity to take into account which features are candidates for matching.

In the following discussion, X is defined to be the value of a feature or measure in a new alert, and Y is the value of that feature or measure in an existing alert class to which it is

being compared. For features that have a single value, the features of a new alert and an existing alert class are defined to be similar if their feature values are equal. That is:

$$SIM(X,Y) = \begin{cases} 1.0, X = Y \\ 0, otherwise \end{cases}$$

The more general case is one in which a feature of an alert includes a list of observed values, and a "hit count" of the number of times each value has been observed. For reasons of normalization, the hit count may be converted to a probability. In this sense, X and Y are lists of feature and probability values, possibly of different lengths. A probability vector describes a pattern over observed categories, where:

$p_X(C)$ = probability of category C in list X,

$p_Y(C)$ = probability of category C in list Y,

$\mathbf{P}_X$ = probability vector over categories observed for X,

$\mathbf{P}_Y$ = probability vector over categories observed for Y.

The notation $C \in X$ is used to denote that category C occurs in list X. The similarity of the two lists is given by:

$$Sim(X,Y) = \frac{\left[\sum_{C \in X \text{ AND } C \in Y} P_X(C) \times P_Y(C)\right]^2}{(\mathbf{P}_X \bullet \mathbf{P}_X)(\mathbf{P}_Y \bullet \mathbf{P}_Y)}$$

If the two lists are the same length, then this measure gives the square of the cosine between the two. This has an intuitive geometric property that is not shared by, say, the dot product as is commonly used in the pattern matching literature.

If the patterns (1, 0), (0, 1), and (0.5, 0.5) are over the same two features, then:

$Sim(\{1,0\},\{0,1\}) = 0$

$Sim(\{0.5,0.5\},\{0,1\}) = Sim(\{0.5,0.5\},\{1,0\}) = 0.5$

In other words, the first two patterns are orthogonal, and the third is halfway in between.

C. Expectation of Similarity

As was discussed above, the nature of an alert may change an expectation of which features of a new alert and an existing alert class should be similar. For example, a syn flood is a type of attack in which the source IP address is typically forged. In this case, similarity

in the source IP address would not be considered when assessing the overall similarity between a new alert triggered by a syn flood attack and an existing alert class.

In a preferred embodiment, the expectation of similarity is a feature-specific number between 0 and 1, with 1 indicating a strong expectation of a match, and 0 indicating that two otherwise similar alerts are likely not similar with respect to a specific feature.

An alert state is preferably represented as a probability vector over likely alert states. For each candidate alert state, each feature has a vector of the same length whose elements are the similarity expectation values (that is, numbers between 0 and 1) given the state. These expectation values may initially be assigned a value of about 0.6 to indicate a medium expectation of similarity, except where it is known that the expectation of similarity is either lower or higher. For example, in an access from a compromised host or a denial of service (DOS) attack, the attacker often spoofs (makes up) the source IP address. Therefore, the expectation values may initially be assigned a value of about 0.1, indicating a low expectation that the source IP address will match that of the attacker.

Based on the alert state, the similarity expectation may be dynamically generated as the weighted sum of the elements of an expectation table, with the weights from the (evolving) alert state distribution. This is itself an expectation in the statistical sense, conditioned on the belief over the present alert state. The similarity expectation is therefore general enough to cover the situation of an unambiguous alert or "call" from a signature-based sensor, in which case the distribution over states is 1.0 at the state corresponding to the alert or call and 0 elsewhere. For example, algebraically for a feature or measure J:

$$E_J = \sum_{i \in \{alert\_states\}} BEL(i) E_J(i)$$

E_J = Expected similarity for measure J, given present state.

$BEL(i)$ =Belief that the attack state is presently i.

$E_J(i)$= Element i of the lookup table of expected similarity for measure J given state i.

Both the new alert and the alert class to which it is being compared each compute the similarity expectation for each feature. Feature similarity and similarity expectation are then combined to form a single value of alert similarity. This is done by combining feature similarities and normalizing by similarity expectation, over the set of common features. For example:

$$SIM(X,Y) = \frac{\sum_j E_j SIM(X_j, Y_j)}{\sum_j E_j}$$

X = Candidate meta alert for matching

Y = New alert

j = Index over the alert features

E_j = Expectation of similarity for feature j

X_j, Y_j = Values for feature j in alerts X and Y, respectively (may be list valued)

A similar definition can be formed by taking products rather than sums above, and then taking the geometric mean. This has some advantages, but can be overly influenced by a single extremely small value.

Where appropriate, similarity expectation may also cover list containment and subnet similarity. For example, an intrusion detection system may generate two alerts, one from a network sensor and one from a host sensor. To decide if the target addresses of these two alerts are similar, the expectation of similarity might be that the target address in the alert generated by the host sensor would be contained in the list of target addresses in the network sensor's alert. In another example, an intermediate expectation of similarity may be generated if target addresses from two alerts do not match but appear to be from the same subnet.

D. Minimum Similarity Requirements

It may be possible to eliminate many of the similarity calculations discussed above if one or more features of an alert class are assigned a minimum similarity value. That is, if a feature of a new alert and the corresponding feature of an existing alert class do not match at some minimum similarity value, the new alert cannot be a member of that alert class. There is therefore no need to perform any additional similarity calculations comparing the new alert to that alert class. Certain features can be required to match exactly or approximately for a new alert to be considered as a candidate for an alert class.

E. Transition Models

When a new alert is encountered, all existing alert classes are preferably passed through their transition models to generate new prior belief states. Transition models attempt to assign a "time value" to alert confidence, as well as try to anticipate the next step in a staged attack. The "time value" is typically modeled as a multiplicative decay factor that reduces alert model confidence slowly over a period of days. The decay period varies by type of attack, and eventually decreases to some background level. This is used to downweight very old alert classes when there are multiple candidates to be compared with a new alert. Transition in time tends to decrease overall suspicion for a specific alert class, as well as spread suspicion over other hypotheses. That is, it tends to make the decision whether to group a new alert with an existing alert class less confident.

The ability to anticipate the next stage in a staged attack is also achieved by a transition in state. Internally, a transition matrix is maintained that expresses the probability of the next state in a staged attack, given the present state. Persistence (the fact that the next step may match the current) is explicitly captured. The transition gives a prior distribution over attack states that is assumed to hold immediately before the next alert is analyzed.

After the transition models generate new prior belief states for their respective alert classes, the similarity expectation for each feature in a new alert is updated. The similarity of the new alert to all existing alert classes is then computed. If none match above some similarity threshold, the new alert is defined to be a new alert class. Otherwise, it is assumed that the new alert is a continuation of the best-matching alert class. By including a list of contributing alerts in the alert class structure, full details of the new alert are retained.

F. Alert Correlation Method

Figure 4 is a flowchart showing how alerts may be aggregated into classes of related alerts in a preferred embodiment. As was discussed above, both alerts and alert classes each have one or more distinguishing features that can be assigned different values. First, a set of potentially similar features shared by a new alert and one or more existing alert classes is identified (step 401). Next, an expectation of similarity between the features of the new alert and features of one or more existing alert classes is either generated or updated (step 403). Next, minimum similarity requirements for the features of the new alert and the features of one or more existing alert classes are generated or updated (step 404). After a comparison

between the new alert and the existing alert class(es) is complete (step 405), the new alert is either associated with the existing alert class that it most closely matches (step 407), or new alert class is defined to include the new alert (step 409).

IV. Alternative Embodiments and Experimental Results

The inventors plan to present various alternative embodiments and experimental results during a conference in late 2001. Relevant portions of a paper submitted for publication during that conference are shown below.

Introduction

In response to attacks and potential attacks against enterprise networks, administrators are increasingly deploying intrusion detection systems (IDSs). These systems monitor hosts, networks, critical files, and so forth, using a variety of signature and probabilistic techniques. See, Porras, P. and Neumann, P. "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," National Information Security Conference, 1997; and Valdes, A. and Skinner, S. "Adaptive, Model-based Monitoring for Cyber Attack Detection", Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, October 2000. The use of such systems has given rise to another difficulty, namely, correlating a potentially large number of alerts from heterogeneous sensors. To the degree that this has been addressed in current systems, heuristic techniques have been used. In this paper, we describe a probabilistic approach to this critical problem.

The intrusion detection community is actively developing standards for the content of alert messages. See, Erlinger, M. and Stanniford, S. "Intrusion Detection Interchange Format," published by the Internet Engineering Task Force. Systems adhering to these evolving standards forward attack descriptions and supporting diagnostic data to an alert management interface (AMI), which may be remotely located and consolidating the reports of numerous sensors. In anticipation of these standards, we are developing systems for alert correlation and ranking.

In Valdes, A. and Skinner, S. "Blue Sensors, Sensor Correlation, and Alert Fusion," Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, October 2000, sensor coupling and an initial approach to probabilistic sensor correlation were introduced. At the time, we demonstrated the utility of the coupled sensors approach for false alarm reduction and improved detection performance. We also introduced a correlation approach based on feature-specific similarity functions. The correlation example cited at that time consisted of a single sensor examining a probe that was distributed in time. Since then, we have introduced some important innovations:

- Capability to comprehend heterogeneous sensors.
- Introduction of the probabilistic minimum match criterion, which prevents spurious matches on less important features from causing the system to correlate alerts. By specifying minimum match of unity, this capability causes the system to function as a rule-based correlation system. Explicitly tolerating slightly imperfect matches provides an important generalization of rule-based correlation methods.
- Hierarchy of sensor correlation: thread, incident, and correlated reports from multistep attack scenarios.

We have also gained considerable experimental experience in both live and simulated environments.

The remainder of this paper is organized as follows. We introduce our approach to alert fusion, using our defined alert template to report alerts from EMERALD and third-party sensors. We introduce notions of feature overlap, similarity, expectation of similarity, and minimum similarity. We show how these are used to achieve a hierarchy of alert correlation defined as inferred thread (within sensor), security incident (between sensor) and correlated attack report (between sensor and attack step). We then present preliminary results from the alerts generated by various EMERALD and third-party monitors.

Sensor Correlation and Alert Fusion

In Valdes, A. and Skinner, S. "Blue Sensors, Sensor Correlation, and Alert Fusion", Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, October 2000, probabilistic methods for sensor correlation were introduced. Specifically, we considered

two closely coupled sensors, a TCP misuse monitor and an asset availability monitor. Both are based on Bayes inference (see, Pearl, J. "Probabilistic Reasoning in Intelligent Systems", Morgan-Kaufmann (1988)), and the former sensor is aware of the state of the latter. Sensor coupling in this fashion is easily expressed in Bayes formalisms; specifically, the coupling is achieved by dynamically modifying priors in the TCP session monitor. The approach demonstrated two important capabilities:

- Failed accesses to services that appear to be invalid are viewed with greater suspicion than other failed accesses. This greatly increases sensitivity against certain stealth probe attacks.
- Certain types of failed access to services that appear "down" are tolerated, so that when a service fails (for malicious or accidental reasons) we do not generate a large number of false alarms.

Having demonstrated the utility of sensor fusion for improved sensitivity and false alarm reduction, we have further explored the general problem of sensor correlation. In particular, we examine the situation of multiple heterogeneous sensors on diverse platforms, considering realistic problems of late reports and clock drift.

Our correlation algorithm considers features as reported in a standard alert template. For each matchable feature, we have defined an appropriate similarity function that returns a value between 0 and 1. New alerts are compared to a list of existing meta alerts. The overall similarity is the weighted average of the feature similarities, with expectation of similarity as the normalizing weights. If the minimum match criterion fails for any feature, the match is rejected regardless of the overall similarity. The new alert is correlated with the most similar meta alert, assuming the similarity is above a specified minimum match threshold. Otherwise, the new alert starts a new meta alert thread.

Meta Alerts and the Alert Template

We have defined an enhanced alert template that enriches standards (such as the Intrusion Detection Interchange Format published by the IETF) with respect to content while avoiding issues of language specifics. Our template includes the concept of alert thread, which is present in all EMERALD monitors and alert management components. Alert reports are part of the same thread if they are from the same sensor and refer to the same

attack, possibly separated in time. Where provided, such as is the case with EMERALD monitors, the thread mechanism causes new reports for an existing thread to be treated as updates, dramatically reducing clutter on a display interface that comprehends threads. For this reason, we identified the capability to reliably infer thread when the sensor does not provide one as a very desirable capability of alert correlation.

We include an "anomaly" field in addition to the "confidence" field used in the evolving IETF standard. We also include arrays to describe in greater detail the target(s) of an attack (for example, the specific ports scanned). Finally, we include fields describing the sensor type and placement, enabling us to identify each unique instantiation of any sensor in the monitored domain.

This template is presently used by a diversity of sensors employing both signature and probabilistic techniques. This includes sensors that we have developed under the EMERALD program, as well as prototype wrappers for ISS RealSecure and Checkpoint Firewall-1. Our early experiments indicate that diverse sensors will be able to fill this template with content that will be more useful to correlation engines than is currently available.

Feature Similarity

Our probabilistic alert fusion approach preferably considers feature overlap, feature similarity, minimum similarity, and expectation of similarity. We maintain a list of "meta alerts" that are possibly composed of several alerts, potentially from heterogeneous sensors. For two alerts (typically a new alert and a meta alert), we begin by identifying features they have in common (feature overlap). Such features include the source of the attack, the target (hosts and ports), the class of the attack, and time information. With each feature, we have a similarity function that returns a number between 0 and 1, with 1 corresponding to a perfect match. Similarity is a feature-specific function that considers such issues as

- How well do two lists overlap (for example, list of targeted ports)?
- Is one observed value contained in the other (for example, is the target port of a denial-of-service (DOS) attack one of the ports that was the target of a recent probe)?
- If two source addresses are different, are they likely to be from the same subnet?

For attack class similarity, we maintain a matrix of similarity between attack classes, with values of unity along the diagonal and off-diagonal values that heuristically express similarity between the corresponding attack classes. We prefer to consider attack classes rather than attack signatures, which are much more specific and numerous but may be erroneously or incompletely reported. For example, in our demonstration environment, we run a variant of mscan that probes certain sensitive ports, that is, it is of the attack class "portsweep." Our host sensors have a specific signature for this attack and call it "mscan." The Bayes sensor trades specificity for generalization capability and has no "mscan" model, but successfully detects this attack as a "portsweep". These reports are considered similar (S = 1) with respect to attack class.

Not all sensors produce all possible identifying features. For example, a host sensor provides process ID, while a network sensor does not. Features not common to both alerts are not considered for the overall similarity match.

The meta alert itself supports the threading concept, so we can visualize composing meta alerts from meta alerts.

Situation-Specific Similarity Expectation

An important innovation we introduce is expectation of similarity. As with similarity, this is also between 0 and 1, and expresses our prior expectations that the feature should match if the two alerts are related, considering the specifics of each. For example, two probes from the same target might scan the same set of ports on different parts of our subnet (so expectation of matching target IP address is low). Also, some attacks such as SYN FLOOD spoof the source address, so we would allow a match with an earlier probe of the same target even if the source does not match (expectation of match for source IP is low).

We now give some examples of how expectation of similarity depends on the situation, that is, the features in the meta alert and the new alert.

If an alert from a sensor has a thread identifier that matches the list of sensor/thread identifiers for some meta alert, the alert is considered a match and fusion is done immediately. In other words, the individual sensor's determination that an alert is an update of or otherwise related to one of its own alerts overrides other considerations of alert similarity.

If the meta alert has received reports from host sensors on different hosts, we do not expect the target host feature to match. If at least one report from a network sensor has contributed to the meta alert and a host sensor alert is received, the expectation of similarity is that the target address of the latter is contained in the target list of the former.

In determining whether an exploit can be plausibly considered the next stage of an attack for which a probe was observed, we expect the target of the exploit (the features host and port) to be contained in the target host and port list of the meta alert.

Some sensors, particularly those that maintain a degree of state, report start and end times for an attack, while others can only timestamp a given alert. The former deal with time intervals, while the latter do not. Similarity in time comprehends overlap of the time intervals in the alerts considered for correlation, as well as the notion of precedence. We preferably do not penalize time similarity too far from unity if the time difference is plausibly due to clock drift.

Deciding whether the attacker is similar is somewhat more involved. In the case of an exact match of originating IP address, similarity is perfect. We assign high similarity if the subnet appears to match. In this way, a meta alert may potentially contain a list of attacker addresses. At this point, we consider similarity based on containment. In addition, if an attacker compromises a host within our network, that host is added to the list of attacker hosts for the meta alert in question. Finally, for attack classes where the attacker's address is likely to be spoofed (for example, the Neptune attack), similarity expectation with respect to attacker address is assigned a low value.

Minimum Similarity

Our correlation component implements not just expectation of similarity (which effectively acts as a weight vector on the features used for similarity matching) but may also enforce situation-specific minimum similarity. Certain features can be required to match exactly (minimum similarity for these is unity) or approximately (minimum similarity is less than unity, but strictly positive) for an alert to be considered as a candidate for fusion with another. Minimum expectation thus expresses necessary but not sufficient conditions for correlation.

The overall similarity between two alerts is zero if any overlapping feature matches at a value less than the minimum similarity for the feature (features for which no minimum similarity is specified are treated as having a minimum similarity of 0). Otherwise, overall similarity is preferably calculated as the weighted average of the similarities of the overlapping features, using the respective expectations of similarity as weights.

By appropriate settings of similarity expectation and minimum similarity, the correlation component achieves the following hierarchy of correlation. The system is composable in that we can deploy multiple instances to obtain correlation at different stages in the hierarchy. For example, we can infer threads (within sensor correlation) and then correlate threaded alerts from heterogeneous sensors into security incidents.

Synthetic Threads: For sensors that do not employ the thread concept, the correlation may synthesize threads by enforcing high minimum expectation similarity on the sensor itself (the thread must come from a single sensor) and the attack class, as well as source and target (IP and ports). We have wrapped the alert messages from a leading commercial sensor and observed that this facility reliably reconstructs threads.

In particular, by placing an aggregator component topologically close to an IDS, the pair is made robust against attacks that cause the IDS itself to flood, as described in the recent National Infrastructure Protection Center advisory 01-004.

Security Incidents: By suppressing minimum expectation of similarity on the sensor identifier, and relaxing expectation of similarity for this feature, we can fuse reports of the same incident from several heterogeneous sensors into a single incident report. In this case, we enforce a moderately high expectation of similarity on the attack class. This is not unity because different sensors may report a different attack class for the same attack. We construct a table of distances between attack classes that expresses which ones are acceptably close. For security incident correlation, we enforce minimum expectations on the source and target of the attack. Using this technique, we have been able to fuse alert reports from commercial and EMERALD sensors into security incident reports.

Correlated Attack Reports: By relaxing the minimum expectation of similarity on the attack class, we are able to reconstruct various steps in a multistage attack. Each stage in an attack may itself be a correlated security incident as described above. In this fashion, it is

possible to recognize a staged attack composed of, for example, a probe followed by an exploit to gain access to an internal machine, and then using that machine to launch an attack against a more critical asset.

Feature Fusion

When the system decides to fuse two alerts, based on aggregate similarity across common features, the fused feature set is a superset of the features of the two alerts. Feature values in fused alerts are typically lists, so alert fusion involves list merging. For example, suppose a probe of certain ports on some range of the protected network matches in terms of the port list with an existing probe that originated from the same attacker subnet, but the target hosts in the prior alert were to a different range of our network. The attacker address list has the new attacker address appended, and the lists of target hosts are merged. The port list matches and is thus unchanged.

Two important features are the sensor and thread identifiers of all the component alerts, so that the operator is always able to examine in detail the alerts that contribute to the meta alert report.

One additional feature is the priority of the meta alert, supported by our template and provided by EMERALD sensors. We are developing a component that estimates criticality based on the assets affected, the type of attack, the likelihood of attack success, and an administrative preference. The aggregator maintains the high water mark for this field. We are investigating approaches whereby the contributing threads are permitted to update their priority downward, computing meta alert priority as the maximum across thread priorities at any given time. This approach would permit downward revision of the meta alert priority.

The features presently considered in the probabilistic correlator component include sensor identification (identifier, location, name), alert thread, incident class, source and target IP lists, target TCP/UDP port lists, source user id, target user id, and time. Computations are only over features that overlap in the alert to be merged and the candidate meta alert into which it is to be merged. Incident signature is used as well, but with a low expectation of similarity as these vary widely across heterogeneous sensors.

If present, a thread identifier from the reporting sensor overrides other match criteria. A new alert that matches the sensor and thread of an existing meta alert is considered an update of the earlier alert.

The correlator first tries to infer a thread by looking for an exact match in sensor identification and incident class and signature. Note that alerts that are inferred to be from the same thread may be separated in time. The system attempts to infer threads even in incident and scenario operational modes.

Next the system checks that all overlapping features match at least at their minimum similarity value. Setting minimum expectation for some features to unity (not normally recommended) causes the system to behave like a heuristic system that requires exact matches on these features. Given that this criterion passes, we compute the overall similarity between the two alerts as follows:

$$SIM(X,Y) = \frac{\sum_j E_j SIM(X_j, Y_j)}{\sum_j E_j}$$

X = Candidate meta alert for matching

Y = New alert

j = Index over the alert features

E_j = Expectation of similarity for feature j

X_j, Y_j = Values for feature j in alerts X and Y, respectively (may be list valued)

Figure 5 is an example of an incident class similarity matrix. Incident class similarity is based on a notion of proximity, which at present is the result of our judgement. The proximity of class A to B reflects how reasonably an attack currently of incident class A may progress to class B. Note that this is not symmetric; we more strongly expect an exploit to follow a probe than the other way around. The incident classes shown in Figure 5 are from the EMERALD 602 message format, but any useful incident classes may be used. Note that some "default" classes such as "invalid" and "action logged" are reasonably proximal to most other classes. This is because the IETF standard does not require a common ontology, and reports from heterogeneous sensors for the same incident may not reliably represent this field. As such, we do not want to reject potential matches based on this field alone. For

operational modes other than thread level aggregation, we do not recommend a high minimum similarity value for this field.

For two alerts that are extremely close in time, it is possible that the alerts may not be in time order. In this case, incident class similarity is the greater of SIM(X, Y) and SIM (Y, X). Mathematically, the similarity computation for incident class can comprehend a discrete call (the alert is from one of the above classes) or a call that is a probability distribution over the above classes (as might result from a meta alert in which the contributing sensors do not agree on the class).

Most other features are potentially list-valued. For lists, the notion of similarity generally expresses the fraction of the smaller list that is contained in the larger. For source IP addresses, similarity also attempts to express the notion that the addresses in question may come from the same subnet.

Time similarity is preferably a step function that drops to 0.5 after one hour. A close match in time is expected only when the system operates in "incident" mode. Thread and scenario aggregation may be over time intervals of days.

RESULTS

Live Data

The following is an example of alert correlation over time, in this case correlating alerts that are components of a stealthy port sweep. The following is an example of one of the contributing alerts. In the interest of space, we do not include all the content that is in the alert and meta alert templates, but limit ourselves to the fields needed to illustrate the result.

```
Thread ID 69156 Class= portsweep   BEL (class) =  0.994 BEL(attack)=
1.000
2001-06-15 17:34:35 from 63.239.148.33 ports 1064 to 1066 duration=
0.000
dest IP aaa.bbb.30.117
3 dest ports: 12345{2} 27374{3} 139
```

This is a probe for three vulnerable ports on a single IP address in the protected network, and is detected by the system described in Valdes, A. and Skinner, S. "Adaptive,

Model-based Monitoring for Cyber Attack Detection," Recent Advances in Intrusion Detection (RAID 2000), Toulouse, France, October 2000. The above is just a single step in a probe that apparently transpired over several days, and resulted in the following correlated meta alert.

```
Meta Alert Thread 248
Source IPs source_IParray: 63.239.148.33 63.239.148.47

Target IPs target_IParray: aaa.bbb.30.117 aaa.bbb.6.232 aaa.bbb.8.31
aaa.bbb.1.166 aaa.bbb.7.118 aaa.bbb.28.83 aaa.bbb.19.121 aaa.bbb.21.130
aaa.bbb.6.194 aaa.bbb.1.114 aaa.bbb.16.150

From  2001-06-15 17:34:35 to  2001-06-21 09:19:57
correlated_alert_priority -1

Ports target_TCP_portarray: 12345{4} 27374{4} 139{3}

Number of threads 10 Threads :69156 71090 76696 84793 86412 87214 119525
124933 125331 126201
Fused: PORT_SCAN
```

We note that we have correlated events from two source addresses that were judged to be sufficiently similar. The attack is quite stealthy, consisting of a small number of attempted connections to single target hosts over a period of days. The list of thread identifiers permits the administrator to examine any of the stages in the attack. In this case, each attack stage is considered a portsweep; if the stages consisted of different attack classes, these would be listed under "Attack steps". Over the three week time period containing this attack, the IDS sensor processed over 200,000 sessions and generated 4439 alerts. The probabilistic correlation system produced 604 meta alerts.

Experimentation Environment

The experimentation environment simulates an electronic commerce site, and provides Web and mail services over the Internet. The protected network is behind a firewall, and is instrumented with several host, network, and protocol monitors. Two machines are visible through the firewall, and two auxiliary machines are used for network monitors and alert management. The firewall blocks all services other than Web, FTP, and mail. We have simulated traffic that appears to come from a number of sources, and an

attacker who executes certain well-known attacks. The background traffic includes legitimate anonymous and nonanonymous FTP use; the latter triggers false alarms in some signature sensors. There are also low-priority nuisance attacks, as well as low-level failed accesses to blocked ports, which trigger firewall log messages.

The attack begins with an MSCAN probe to the two machines visible through the firewall. Signature-based sensors on the two machines detect this and set the attack code for mscan in their alert report. The Bayes network sensor detects the attack as of the class "portsweep". The target port lists for these alerts match, as does the attacker address. Similarity with respect to the target address depends on the order in which the alerts arrive (the order at which the alerts arrive is not deterministic as the sensors are distributed). Until the network sensor's alert is received, we do not expect target addresses to match for distinct sensors, since a host sensor can typically report only its own address as a target. We expect the network sensor's target list to contain the targets seen by host sensors (tolerating imperfect matches due to arrival order), and we expect the target address of subsequent host alerts to be contained in the meta alert's target host list. Therefore, regardless of arrival order, these alerts are acceptably similar with respect to target host.

The attacker next uses a CGI exploit to obtain the password file from the Web server. This exploit is detected by the host sensor. The next stage of the attack is a buffer overflow to gain access on the host providing mail service, detected by an EMERALD host sensor as well as by ISS.

Although telnet through the firewall is blocked, the attacker may telnet (using a password obtained from the earlier exploit) from the compromised internal host to the critical host, and he now does so. On that host, he uses another overflow attack to obtain root access. He is now free to change Web content.

Figure 6 shows the EMERALD alert console for this scenario, containing alert reports for the above attack steps as well as alerts due to false alarms and nuisance attacks. It is not very informative, except to show that more than 1000 alerts are received for this scenario, which lasts approximately 15 minutes, and the critical attack is effectively buried.

The majority of the alerts are from nonanonymous FTP write operations, which are allowed but trigger alerts from a commercial sensor.

Figure 7 shows the result of inferring synthetic threads for sensors that do not support the threading concept. For alert thread inference, we preferably set the minimum match for sensor identifier fields to unity, and require a high match for attacker and target parameters as well; we do not require a minimum similarity in the time field to allow inference of threads that occur slowly over time. Most of the alerts are threaded into sessions according to originating IP address, and the total number of alerts reported is reduced to about sixty. We have highlighted an alert, which is one of the nuisance TCP sweep attacks in the background traffic.

For attack incident matching, we must look across sensors to determine if several alerts refer to the same security incident. We preferably suppress minimum similarity for sensors, set a moderately high similarity for attacker and target identifiers, and set a moderate minimum for time (to allow for clock drift and sensor response time differences). We may also require a match in attack class, making allowances for sensors that report different attack classes for a given attack sequence.

Figure 8 shows the result of incident matching for the attack scenario. Some of the critical steps in the attack reported by several sensors now result in a single alert. For example, the highlighted buffer overflow attack is reported by an EMERALD sensor as well as by ISS. These are fused into a single correlated incident report. The two mscan reports from EMERALD host sensors and the portscan report from the EMERALD Bayes TCP sensor are also fused into a single incident.

The final step in correlation is an attempt to reconstruct the attack scenario. We now relax attack class similarity to consider what attack steps might follow the present step. For example, it is reasonable to follow a probe with an exploit to a target "contained" in the probe. Also, if the meta alert includes a step indicating a likely root compromise, the host in question is added to the attacker's assets (that is, it is both a victim and now a potential attacker as well). In Figure 9, the correlated attack report correctly lists as attacker assets his originating host, an internal host compromised at an intermediate step, and the spoofed address of the SYN flood stage. In this fashion, we have successfully combined both the external and internal threads of the attack scenario.

It is worth pointing out that the EMERALD sensors in this experiment provide response directives which, if followed, would stop the attack in progress.

We have adapted and extended notions from the field of multisensor data fusion for alert correlation. The extensions are principally in the area of generalizing feature similarity functions to comprehend observables in the intrusion detection domain. The approach has the ability to fuse alerts for which the match is good but not perfect. The method considers appropriate fields in alert reports as features for a multivariate matching algorithm. Features that overlap are considered for the overall similarity calculation. For each matchable feature, we have defined an appropriate similarity function with range 0 (mismatch) to 1 (perfect match). Depending on the situation, we incorporate expectation of match values (which are used to compute a weighted average of similarity over the overlapping features), as well as a minimum match specification that unconditionally rejects a match if any feature fails to match at the minimum specified value. For each new alert, we compute similarity to existing meta alerts, and merge the new alert with the best matching meta alert, as long as the match passes a threshold value. We realize a reduction of one-half to two-thirds in alert volume in a live environment, and approach a fiftyfold reduction in alert volume in a simulated attack scenario.

WHAT IS CLAIMED IS: